AFRL-RI-RS-TR-2008-244
Final Technical Report
September 2008

THE COMMON MANET FRAMEWORK

BAE Systems

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2008-244 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
KURT TURCK Work Unit Manager	WARREN H. DEBANY, Jr. Technical Advisor, Information Grid Division Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
SEP 2008	Final	MAY 07 – MAY 08

4. TITLE AND SUBTITLE

THE COMMON MANET FRAMEWORK

5a. CONTRACT NUMBER
FA8750-07-C-0154

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Gregory Frazier

5d. PROJECT NUMBER
558J

5e. TASK NUMBER
MA

5f. WORK UNIT NUMBER
NT

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

BAE Systems National Security Solutions, Inc.
BAE Systems Advanced Information Technologies
6 New England Executive Park
Burlington, MA 01803-5012

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RIGC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2008-244

12. DISTRIBUTION AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2008-0017

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This effort researched, developed and demonstrated a Common Mobile Ad-hoc Networking (MANET) framework (CMF) to provide a uniform abstraction for wireless discovery, communication, and routing of network traffic over heterogeneous transmission technologies. The application program interface (API) supports the development of MANET-appropriate Internet Protocol (IP)-based applications and also provides observability and controlability interfaces to the CMF. The research uses two complementary algorithms: one uses application behavior to identify initial routes (Application Informed Routing –AIR) and the other maintains and improves those routes over time (Dynamic Virtual Circuits – DVCs). In combination, these algorithms allow wireless routing to scale O (1) with respect to MANET size. The effort enables efficient routing for arbitrary sized MANETs. The Application Program Interface and Transmission Layer Interface allow multiple transmission technologies to interoperate, shielding both the applications and system administrators from having to address the heterogeneity.

15. SUBJECT TERMS

Dynamic node discovery, Application informed routing, Tactical Internet Protocol (IP), (IP) Radio, Airborne Networking, MANET, Dynamic Virtual Circuits (DVC).

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Kurt A. Turck
U	U	U	UU	25	19b. TELEPHONE NUMBER (*Include area code*) 315 330-4379

Table of Contents

List of Figures

1 Introduction

We perform research in large-scale distributed autonomic systems and computer system defense, and we work with Mobile Ad-hoc Network (MANET) systems and protocols. Over the past two years we have performed research on cyber security for MANET systems for DARPA STO[1], conducting a wide range of experiments on a daily basis on our 72-node MANET test bed. Protocols exercised on this MANET included: Optimized Link State Routing Protocol (OLSR [6] – a unicast routing protocol for MANETs specified in RFC 3626); Simplified Multicast Forwarding (SMF – a multicast routing protocol that operates on top of OLSR and is described in an IETF draft specification [7]); Simplified Link State Routing (SLSR – a unicast protocol), ROM and ODMRP [8] (multicast protocols); and SPIN [25] (an efficient broadcast protocol designed for sensor networks). The MANET was exercised with five scripted tactical applications developed by Naval Research Laboratory (NRL).

One of the fascinating, though not entirely unexpected, aspects of this work was how poorly both the protocols and the applications scaled in the MANET. The overall system was barely able to scale to 72 nodes and did so only by exploiting some rather optimistic movement and radio assumptions. Our experience is not unique; similar observations regarding MANET scalability have been made in other contexts [1,2,3,29]. Over the course of our DARPA STO work we developed three insights as to why efforts at MANET development have fallen short:

1. *The set of design patterns that are viable on MANETs are different from those commonly in use on wired networks.*

MANETs impose restrictions on application architectures that wired networks do not. The amount of bandwidth available to nodes in a MANET diminishes as the MANET grows, encouraging geographic locality. In wired networks, long-haul links have large amounts of available bandwidth and geographic locality is rarely considered. The composition of MANET networks is dynamic as nodes move in and out of radio range of each other; robust MANET applications use discovery and self-organization to form distributed collaborations. Wired networks, on the other hand, are relatively stable, and static configuration files are often used as the means for application organization. Finally, route discovery is expensive in the MANET; this expense is ameliorated by temporal locality (i.e., using the route for an extended period of time). In wired networks, there are no cost advantages to sending consecutive packets to the same destination. We contend that there are MANET-appropriate design patterns that help developers codify the characteristics of discovery, and geographical and temporal locality in their applications.

2. *A broad range of applications can be effectively implemented using MANET-appropriate design patterns*

While the design patterns described above are not necessary for wired-network applications, they are often used to improve their performance or make them more robust. In other words, discovery/self-organization, geographic locality and temporal locality are concepts that are incorporated into middleware and discovery / deployment services that are currently in use in wired systems and could be used (with some modification) in MANETs.

3. *MANET infrastructure design should focus on supporting applications that implement scaleable MANET-appropriate design patterns.*

We have asserted that applications must possess specific properties in order to be viable in a tactical MANET and that a wide variety of applications can be implemented such that they possess those properties. This implies that MANET design should focus on supporting applications that possess these properties and not focus on supporting applications that do not possess those properties.

[1] The Defense against Cyber Attack (DCA) MANET Program, first under Dr. Anup Ghosh and then Mr. Chris Ramming, now transitioning to Future Combat Systems Lead Systems Integrator (FCS LSI).

A very expensive feature that MANETs currently implement is *universal routing* – routing for a *logically* fully connected network. This is the routing environment that is available on the Internet. In a large network, universal routing mostly goes unutilized; if one were to randomly select two of the ~10^8 computers on the Internet, chances are very good that they will *never* communicate. Similarly, in a large MANET hosting well-architected applications, most communications will take place between a small set of the possible node pairs (geographic and/or temporal locality). This is a good thing – if a process were to regularly communicate with most or all of the other nodes in the MANET, the MANET would fail as it scaled, irrespective of how efficiently the routing protocol supports this behavior. Paradoxically, the goal of *every* general-purpose MANET routing protocol[2] that we are aware of is to provide universal routing – an attempt to recreate the wired networking environment.

On the other hand, to date very little consideration has been taken as to what features or functionality are necessary to support the MANET-appropriate characteristics described above. Previous efforts have offered the standard network APIs without enhancement. In other words, there has been little thought given to what applications might require from the MANET in order to implement MANET-appropriate design patterns.

Note there is design requirement implied by the paragraph above: MANETs must violate the isolation principles espoused in the ISO network layers model. Conventional networks hide the state of routing layers from each other, leveraging modularity to simplify network construction.

The Common MANET Framework (CMF) was born from these insights. It has three components:

- A scalable, efficient communication infrastructure that does not depend on universal routing, but instead leverages the discovery/self-organizing activities inherent to MANET applications to efficiently route data in the MANET;
- The APIs and services that provide applications with the functionality that they require in order to implement MANET-appropriate design patterns (the green components);
- An open transmission layer interface (TLI) that allows the CMF to be bound to a variety of communication technologies, supporting the deployment of heterogeneous MANETs (i.e., MANETs that integrate multiple trans-mission technologies into a coherent network such that the heterogeneity is not apparent at the IP level) (the purple component).

§1.1 provides an in-depth discussion of the CMF, describing the rationale for the technologies proposed and comparing them to published MANET research. §1.2 is the technical program summary – a compact description of the CMF from top to bottom (a summary of §1.1). In §1.3 we present the risk factors present in this program. Finally, §1.4 is a listing of the publications referenced in this proposal.

2 Properties of (Design Patterns for) MANET Applications

We identified three properties that must be exhibited by MANET applications:.

Property 1: Geographic (topological) locality of communication. For a single hop, the bandwidth made available by a radio can be comparable to that provided by Ethernet. 802.11g specifies a 54 Mbits per second transmission rate – this is comparable to 100 Mbit Ethernet. The difference between wired and wireless arises as the distance between communicating nodes increases and their traffic must be forwarded through multiple hops. Wired networks have high-bandwidth backbones to carry long-distance traffic. In contrast, the amount of bandwidth across a MANET available to each node decreases as the MANET grows (the MANET grows O(n), the bandwidth across a bisection of the MANET grows $O(n^{1/2})$ in the most optimal of circumstances). While some MANET applications will have to perform long-distance communications, they must do so in a controlled fashion. In general, MANET applications must display *geographic locality – the tendency to communicate with nodes that are nearby in the MANET topology*.

[2] There are sensor networks that do not implement universal routing – but these are not general-purpose.

Property 2: Temporal locality of communications. Another difference between wired networks and MANETs is that the topology and inter-node routes of a wired network are relatively stable, while the topology and routes in a MANET are dynamic. This suggests that, in a MANET, there is some cost associated with discovering how to route packets between two hosts. This cost is offset by the benefit derived from the successful communication between those hosts. We posit that *temporal locality – the tendency for two nodes that are communicating to continue to communicate* – is a property that MANET applications must possess, especially when performing long-distance communication in the MANET.

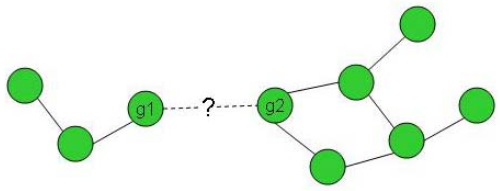


Figure 1: MANET Partitioning.

Property 3: Discovery and self-organization of applications. A third difference between MANETs and wired networks is the tendency for MANETs to partition into multiple separate networks and/or join into a single network, and for new participants to enter the network (possibly without a priori configuration) and for current participants to leave the network. When the nodes g1 and g2 in Figure 1 move outside of each other's radio range, what was a single MANET becomes two independent networks, and when they move back into each other's range, the two MANETs become a single network. Given node mobility, the dynamic nature of radio communications, and the abruptness with which participants enter and leave the battlespace, a computer will find itself networked with an unpredictable set of nodes in the MANET. So, for applications to successfully operate, the processes that comprise the application must discover and make use of whatever computing resources are available on the network at any particular point in time. So, *discovery and self-organization* is a design pattern that is **necessary**[3] for successful MANET applications.

Of course, discovery and self-organization is a useful design pattern in wired networks, and there are commercial products and standards that provide it. Publish-and-subscribe infrastructures (e.g., JMS), service brokers (Sun's Jini, the CORBA Trading Object Service) and lookup (name) services (CORBA Name Service, LDAP, the Java RMI Registry) all fall under this heading. So, while an application must incorporate discovery and self-organization to operate in the MANET, the discovery functionality can be external to the application. Note that the discovery service is itself an application that must be implemented in a MANET appropriate fashion (i.e., incorporate the design patterns and properties just discussed, *including discovery and self-organization*).

We draw two conclusions from this analysis.

- If applications must possess these design patterns in order to function in a MANET, the MANET should provide the functionality necessary for the design patterns to be implemented. Thus, if applications must incorporate discovery in order to operate in the MANET, then the MANET should provide support for discovery.
- The MANET should concentrate on supporting large-scale applications that possess these design patterns and should avoid costly features intended to support applications that do not possess them.

The CMF takes these conclusions one step further – it *depends* on the fact that applications will possess these design patterns (particularly discovery and self-organization).

[3] One can imagine an application that is based entirely on one-hop broadcasts that does not utilize a discovery/self-organization pattern as we have described. However, it would also not use MANET routing. So, while it is important for the CMF to not preclude simplistic application structures such as this, they are not under consideration in our discussion of MANET design patterns that must be supported.

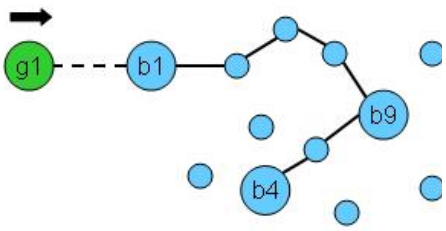


Figure 2: g1 discovers b9 via b1, and then discovers b4 via b9.

3 Scalable Routing

The central idea behind the CMF's novel approach to routing is to establish what might be described as a symbiotic relationship with the applications that use the CMF. On the one hand, the CMF provides the functions (and APIs) necessary for nodes (and the processes on those nodes) to discover one another and self-organize. On the other hand, the applications (or middleware representing the applications), by virtue of using the CMF discovery APIs, provide the CMF with the information it needs to route packets to their destinations. We call this Application Informed Routing (AIR) and note that it is a *revolutionary technique for routing packets in the MANET, utterly different from the proactive, reactive, geographic, and even hybrid MANET routing protocols that represent the current state of the art.*

The inter-node routes revealed by AIR are unlikely to be optimal. And even if they were, the MANET is a dynamic environment; node movement and network congestion cause routes to change over time. The CMF will implement Dynamic Virtual Circuits (DVCs) [24], a technique for maintaining and optimizing routes within the MANET. Virtual circuits provide "route memory", allowing subsequent packets in a flow to follow the same route as the first. By making the virtual circuits *dynamic*, we give the nodes between the source and the destination the ability to change the route. So, even if AIR provides an inferior route, the resulting DVC will improve the route over time. DVCs provide: a reliable packet stream between processes; continuous optimization and maintenance of communication flows in the MANET; a record of node connectivity that will supplement AIR as a means for discovering routes; a basis for reporting network statistics to self-organizing applications; and perhaps most importantly, a means for discovering the loss of network resources due to MANET partitioning. The following subsections describe AIR and DVCs in more detail.

3.1 Application Informed Routing – AIR

Given that AIR requires application activity in order to generate routing information, it is helpful to have an example application to illustrate how AIR will work. Consider a self-organizing lookup service, where a group of nodes select one of their members to be the lookup server, and the other nodes in the group advertise services and/or look up services on that server. When the MANET grows too large for a single lookup server, the lookup application self-organizes into a more complex system, with multiple nodes hosting lookup servers and the lookup client applications organizing themselves into groups by selecting which nearby lookup server to subscribe to.

In order to illustrate AIR's functionality, we will focus on the sequence of events from when a node newly joins an established group to the time that it successfully establishes a connection to the lookup server. Figure 2 depicts the example. The blue nodes have self-organized into a group with b9 as the lookup server. Every node in the group has a route (a DVC) established to b9, and b9 has a route to each host (routes from b1 and b4 to b9 are shown). The example begins when a new node (g1) comes within radio range of the blue group.

The discovery process is bootstrapped by one-hop neighbor discovery; processes can register to be notified of changes in the one-hop neighborhood. So, when g1 comes within radio range of b1, the lookup application on g1 is notified of b1's presence. The lookup application responds to the notification by sending a packet to the lookup application on b1 asking who its lookup server is. Given that b1 is g1's immediate neighbor, routing the message is trivial – the message is sent to b1.

The lookup application on b1 responds that b9 is its lookup server. Given that this is a *discovery* (g1 is discovering b9 via b1), b1 uses the CMF discovery API to send this information to g1. *Note that it is the use of a structured API for transmitting discovery information between processes that gives AIR visibility into the applications' discovery activities.* Now g1 sends a message to b9 to initiate use of the lookup

service. This is where AIR comes into play. As the route setup packet from g1 to b9 is processed by the CMF on g1, AIR knows that b1 was the node that informed g1 of b9, and so routes the packet to b9 via b1. When the packet arrives at b1, the routing algorithm there determines that a route exists between b1 and b9, and the packet follows the existing route to b9. Thus, the application discovery process is used by AIR to find a route between two nodes – g1 and b9 – for which a route did not previously exist. And, if g1 uses the lookup service on b9, and b9 tells g1 about a server residing on b4 (again using the discovery API), then when g1 establishes a connection to b4, AIR will route the packet via b9 in the knowledge that b9 has a route to b4. As has already been acknowledged, a route from g1 to b4 that traverses b9 will be sub-optimal; AIR depends on the resulting DVC to improve the route.

There are three addenda to this example. The first is that AIR is an incredibly simple algorithm. Second, for clarity we ignored the fact that intermediate nodes may already have routes to the destination. In the example, g1 establishes a route to b4 by way of b9 – but if b1 already has a route to b4, it will use that route instead of continuing to b9. Third, while the lookup service must incorporate the CMF discovery API, applications using the lookup service would not need to do so. More generally, applications that use middleware for discovery and self-organization will be able to continue to use the standard interface provided by the middleware, but the middleware will have to incorporate the CMF discovery APIs.

3.2 Dynamic Virtual Circuits (DVCs)

DVCs are a mechanism for maintaining and improving routes in store-and-forward networks[4] [24]. The need for such a mechanism is driven by the fact that the route between two nodes will change over time. While AIR is satisfactory as the mechanism to discover a route between nodes, the correct operation of AIR requires that routes, once established, be maintained. Universal routing provides this functionality by broadcasting topology changes to the entire MANET (proactive routing), flooding route-discovery control packets through the entire MANET (reactive routing), or propagating geographic locations and/or location queries through the entire MANET (geographic routing). DVCs allow us to restrict the distribution of topological changes to only those nodes that require the knowledge.

Virtual circuits are a well known technique for routing packets in a store-and-forward network. The virtual circuit is a series of hops from the source to the destination. Once the circuit is established, packets need only carry the identifier of the virtual circuit, as each node maintains a table that maps circuit identifiers to their next hop. The use of virtual circuits in the MANET poses two challenges. The first is how to establish the circuit; we have already discussed the use of AIR for this purpose. The second challenge is that the dynamic environment of the MANET will cause routes to change over time. We address this challenge by making the virtual circuits *dynamic*.

The dynamic component of DVCs is the fact that any node on the circuit can autonomously change the path of the circuit. Three aspects of the DVC design allow this to work. First, DVCs are one-way circuits. While control data passes in both directions, data packets flow in a single direction, and all DVC route changes (setup and teardown events) are restricted to going in that same direction. Second, every node in the MANET knows its two-hop neighborhood (the nodes that are within a two-hop distance and the links between those nodes). Third, every node on a circuit knows the identity of every other node on the circuit. We will explain each of these in more detail and then offer examples of DVC maintenance and improvement.

Restricting DVCs to routing packets in a single direction allows the nodes that comprise the circuit to act independently of each other with regard to routing decisions. In other words, it eliminates the need to coordinate routing decisions across multiple nodes in the circuit, dramatically simplifying both the circuit maintenance algorithm and the DVC state space. The key simplifying elements are:

[4] DVCs were originally intended for use in scalable multi-computers. The DVC algorithm presented here has been modified from the original to address CMF requirements.

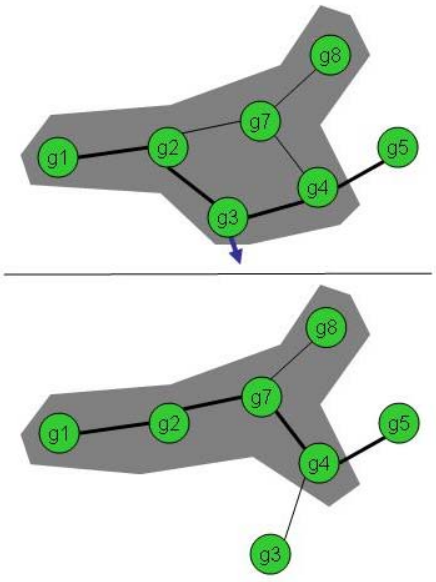


Figure 3: A DVC repairing a route.

a) Packets are all pushed in front of a teardown event. There is never a circumstance where a circuit teardown removes a link that a data packet then needs to traverse.

b) Since the packets are all going in the same direction, they can be assigned identifiers that are monotonically increasing, and all circuit establishment and teardown events can use the subsequent packet ID as their own identifier. So, when a node must handle a complicated DVC event sequence (e.g., receiving setup and teardown control packets for the same circuit from different neighbors), the node knows in what order to process the events.

Admittedly, TCP is always bidirectional, and thus requires a reciprocal pair of DVCs, so one might consider the duplication of control logic over two separate DVCs to be a waste of resources. However, analysis of the DVC algorithm suggests that DVC control will incur minimal computation and communication overhead, and the use of separate circuits in each direction gives the CMF the ability to use different routes for the two circuits, dispersing the interprocess traffic over a wider set of nodes. The bottom line is that dynamically adjusting the route of a bidirectional circuit is challenging; the use of one-way circuits *dramatically* simplifies the problem.

The second component of the CMF design that supports DVC operation is that every node has an awareness of its two-hop neighborhood. This knowledge of its local neighborhood ensures that nodes can repair circuits when individual links are broken and can make incremental circuit improvements. The selection of two hops as the radius of local topology awareness is based on two factors. First, it is the smallest radius that can possibly support repair and incremental route improvement. Second, the algorithm for two-hop neighborhood awareness is trivially simple: every node periodically shares its list of one-hop neighbors with each of its one-hop neighbors. Since the cost of neighborhood awareness increases $O(r^2)$ with r being the radius of the awareness, there is great incentive in making DVCs work without increasing the radius beyond two. Note that two-hop topology awareness provides an alternative to AIR for routing to destinations two hops away or less.

The third design decision integral to DVC success is that every node in a circuit is aware of every other node in the circuit. While a circuit is being established using AIR, each successive node is made aware of all of the preceding nodes on the circuit. Once the destination is reached, an acknowledgement packet makes the initiating node aware that the DVC is ready for use and makes every node on the circuit aware of the nodes that follow it on the circuit. Thereafter, any modification to the DVC generates control packets that go in both directions from the point of change, updating each node's awareness of the DVC's composition. This allows any node in the circuit to recognize an alternate route to an intermediate node in the circuit.

Figure 4 shows how a node uses local topology awareness and the identity of the other nodes in a DVC to repair a route. The figure depicts a MANET in which there is a flow from g1 to g5 that passes through g2, g3 and g4. In the figure, g2's awareness of its two-hop neighborhood is depicted by the gray

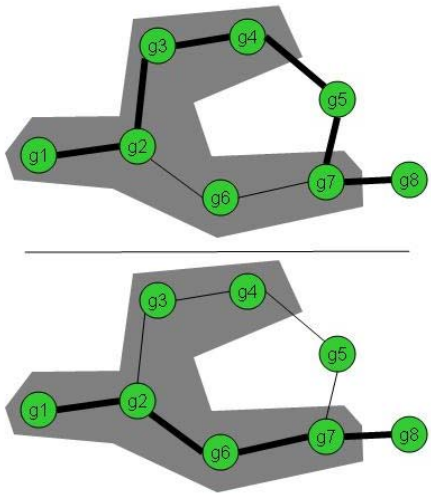


Figure 4: Proactive DVC route improvement.

background – g2 is aware of every node and link in that region. So, if the link between g2 and g3 is broken, g2 can repair the route by creating a new path via g7 to g4. g3 will also detect that the link to g2 is broken, and will (after a timeout) initiate a DVC teardown that will propagate only to g4 (since g4 has already received the rerouted circuit via g7, it knows to drop the teardown request).

Note that DVCs do not require that a path be broken before a new path is created. In Figure 3, g2 uses its two-hop neighborhood (gray background) plus its knowledge of the identities of all of the nodes in the DVC to detect an inefficiency in the route. While g7 is four hops away from g2 in the DVC, it is physically two hops away. So, g2 establishes a new path via g6, and then sends a teardown command to g3 that terminates at g7.

There is an additional benefit to keeping every node in the DVC informed of the DVC's composition – by providing this knowledge, DVCs supplement AIR and two-hop neighborhood awareness as means for discovering routes.

3.3 CMF Routing Summary

To quickly summarize, routing in the CMF is accomplished by two interrelated algorithms. AIR leverages application discovery to find routes between nodes. DVCs provide both memory of those routes over the course of an interprocess conversation *and* a mechanism to maintain and/or improve that route in the dynamic network. There are three aspects of CMF routing that allow it to scale where conventional routing protocols have failed: first, there is no network traffic generated by the routing algorithm to discover routes; second, route information is only propagated as far as the distributed applications require it to be propagated; and third, detailed topology information (the identity of the nodes in a route) is only provided along that route (as opposed to being broadcast to the network). The analysis that we have performed to date indicate that it will only be application behavior (and application artifacts such as network congestion) that limits MANET scalability; the CMF routing algorithm should never be the impediment.

4 The CMF Application Program Interface

The CMF Application Program Interface (API) is designed to provide access to CMF functionality while minimizing the impact of the transition from programming for the wired environment to programming for the MANET. Experiments that we have performed developing MANET applications have shaped our notion of the correct approach to API development. Key elements of our approach are:

- Support for MANET design patterns.

  Of the three design patterns we identified for MANET applications, two profit by direct support. Geographic locality cannot be implemented unless a process can collect metrics regarding the "distance"[5] between it and the processes with which it is interacting, and discovery is greatly facilitated by direct support. These are not concepts that were considered in the design of the Windows and UNIX operating systems. So, functionality must be added to the communication API.

- Adherence to existing interface specifications, allowing applications to be oblivious to the CMF where possible.

  We in no way modified the socket APIs. The CMF APIs are orthogonal to the transmission of individual packets. Where an API does cause data to be transmitted (e.g., `cmf_sendDiscovery`), the transmitted data does not traverse a socket (e.g., the data is transmitted below the IP abstraction layer).

- An ability to be implemented in wired environments with minimal effort.

[5] Distance here can simply mean physical distance (number of hops), but it can also include metrics such as transmission quality and congestion. Geographic locality is a special case of topological locality.

Most developers of MANET applications will first build and test their application in a conventional wired environment. Our plan is to implement the CMF the wired environment prior to being ported to a wireless environment. While we are doing this for programmatic reasons, a positive ramification of this approach is that developers will be able to code and test in a wired environment before moving their application to real or emulated MANETs for system testing.

Please see the file `cmf_X.X.X/include/cmf.h` in the CMF code delivery for a listing of the API.

5 The CMF Transmission Layer Interface

There are three reasons to define a standard interface between the transmission layer (the layer responsible for transmitting packets between nodes) and the CMF. The first is to facilitate the porting of a CMF implementation between transmission technologies. Just in the course of executing this proposal, we anticipate running the CMF on a wired LAN, a MANET test bed with emulated radios, on Linux laptops with 802.11g-compliant radios, and on Linux laptops with software defined radios. By establishing an interface with well-understood semantics, we will facilitate transition of the CMF technology to multiple environments.

The second reason to specify a Transmission Layer Interface (TLI) is that military MANETs will be comprised of heterogeneous transmission technologies. A fielded military MANET may exist for years and will be comprised of nodes from many military units. It is naïve to think that they will all come to the battle with identical communication equipment. Certainly, for two nodes to directly communicate with one another, they must have compatible radios. However, individual nodes with multiple radios or nodes with different radio technologies linked by a wired LAN can act as gateways between transmission technologies. Given the CMF routing algorithms described above and the interface specification described here, the CMF can transparently utilize such gateways to incorporate heterogeneous technologies in a single MANET. Not only will the application/IP layer be unaware of the heterogeneous nature of the transmission layer, *no portion of the CMF above the TLI will require modification to address specific transmission technologies.*

The third reason to specify a TLI is that the majority of the radio technology currently deployed by the U.S. military is not designed for MANET operation. Specifically, Tactical Data Information Link (TADIL) radios are heavily used by all branches of the military. The most common TADIL standards are Link-11 (TADIL A and B) and Link-16 (TADIL J). If some or all of these are to be incorporated into a heterogeneous MANET, there will have to be a well-defined structure for doing so.[6]

The CMF implementation has two bindings implemented for it. One is a multicast binding – outgoing packets are wrapped in multicast UDP packets. The second is a UDP binding that can either be a unicast or a one-hop broadcast, depending on configuration.

6 Lessons Learned

6.1 Design

6.1.1 Reliable Packet Delivery

It is common for MANET developers to ignore the requirement for reliable packet delivery. It is impossible to guarantee that a packet will reach is destination, even in wired networks. To make this concept explicit, protocol developers use the term *best effort delivery*. Unfortunately, many MANET developers have used this concept as license to ignore the requirement to deliver as many packets as possible. They are abetted in ignoring this requirement by three factors. The first is that most MANET

[6] We note that BAE NES – a sister organization of AIT – has implemented a special-purpose MANET over Link-16 radios called FAST [32].

performance evaluations focus on *goodput* (the rate at which packets reach their destinations) as opposed to tracking the fraction of packets delivered to their destination. The second is that performance evaluations are typically performed using sham applications that send their packets to randomly-selected destinations. Without locality, the MANET will be overwhelmed by traffic and will unavoidably drop many packets. Third, there is a common misconception that dropping packets is unavoidable in MANETs, and so there is no point in trying terribly hard to deliver every packet.

There are three consequences to the abandonment of reliable packet delivery in MANETs. The first is that folks have concluded that TCP does not work on MANETs. And, a corollary, that reliable packet delivery on MANETs is the province of the applications. The third fallout, which naturally stems from the previous two, is that reliable packet delivery is often attempted in MANETs by sending redundant data. For instance, if one has a video stream and requires 8 frames per second, one sends 24 frames per second in the hope that at least 8 frames per second arrive at the destination.

These consequences have an insidious influence on MANETs. One notes that, in wired networks, the vast majority of the application protocols operate over TCP. It is very hard to write an application that does not care whether its messages are received by the destination. Even those applications that run over UDP (e.g., a video feed) have an implicit requirement that most of the packets will arrive at their destination. The middleware/application layer is in no better position to achieve reliable packet delivery than the network layer, and often is in a worse position. And greedy approaches to reliable packet delivery (sending more packets than are needed in the hope that enough packets reach their destination) is both self-defeating (when every node pursues this strategy, congestion problems are exacerbated, and network congestion is the bane of MANETs) and infeasible (a greedy approach assumes that packets are discarded with a uniform distribution, when in fact the dropping of packets is bursty).

So, a design goal of the CMF is to provide reliable packet delivery over MANETs. To accomplish this, the CMF incorporates three inter-related features. First, it uses data packets to detect the loss of communication neighbors, by having every transmitted packet acknowledged. When multiple consecutive transmission attempts are unacknowledged, the CMF assumes that the neighbor is no longer present and commences a re-routing event. This avoids the situation where multiple seconds go by with a node transmitting to a neighbor, not knowing that that neighbor is no longer there. Second, the CMF will attempt to transmit a packet multiple times; packets are not removed from the buffer until a positive acknowledgement is received or the maximum number of transmission attempts has been made. This allows the MANET to continue to operate over less-than-perfect connections. Third, when a DVC is re-routed, the DVC construction packet bypasses any IP or discovery packets in the queue that are on the same DVC, and those packets have their one-hop neighbors modified to the new route.

The CMF has three parameters that control this behavior (all found in `cmf_X.X.X/include/cmf_build_parameters.h`): the number of consecutive unacknowledged packets before the CMF decides that the neighbor is not there (`MAXIMUM_OUTSTANDING_PACKETS`), the number of attempts to transmit a packet before it is dropped (`MAXIMUM_NUMBER_OF_TRIES`), and the acknowledgement timeout (the time between transmission attempts for a packet) (`ACKNOWLEDGEMENT_TIMEOUT`).

It is difficult to isolate the impact of these three mechanisms on the performance of the CMF. For example, one might argue that *any* MANET protocol would benefit from an acknowledgement/re-transmission scheme. MANE (the NRL emulation environment) supports two models of radio signal propagation: an r^3 model (where the transmission failure probability escalates as the distance between two nodes approaches the transmission range); and a step function (failure probability transitions from zero to one at the transmission range). I used the latter model in my comparisons between the AIR and OLSR to avoid incorporating the benefits of repeated packet transmission, which one might argue are not an inherent feature of AIR could as easily be applied to OLSR.

Mind, this violates one of the tenets of the original CMF proposal – that one must evaluate the MANET as a system, and that one of the inherent failures of MANET research to date has been the insistence on taking a reductionist approach, where individual components are modified while leaving the rest of the

system constant (e.g., taking a wire-based system, modifying one component, and seeing how well it supports wireless). As will be seen in this report, this is one of several points where I violated this tenet, with less than satisfactory results.

6.1.2 Buffer Size

In the delivered CMF, the buffer size is set to 20 packets. This is large – too large. The problem is that full buffers are a disaster. When a buffer becomes full, the buffers of the neighboring nodes start to fill up. This is the downside of a strategy that emphasizes reliable packet delivery.

The CMF does not require a 20-packet buffer, and my suspicion (not confirmed by performance analysis) is that it will perform very well with a smaller buffer. There were three factors during the development of the CMF that led to the use of 20-packet buffers.

1. During the debugging process, I occasionally had problems that caused bursts of packets. If those bursts caused packets to be dropped, it made it more difficult to debug the system. So, a large buffer assisted with the debugging process.
2. During the debugging process, I often utilized what we call the "string of pearls" topology—a series of nodes where each node has exactly two neighbors (except for the endpoints, which have only one neighbor). This topology creates congestion across the middle links, if the nodes are selecting communication partners from a random distribution over all nodes.
3. In the emulation environments that I have used to exercise it, the CMF has consistently suffered from intermittent link drops. My initial strategy was to immediately send packets to update the state of the DVCs that traversed the down link. This caused a packet burst that quickly filled smaller buffers. I then changed the strategy, going for an immediate update only of recently-used DVCs. I finally settled on an approach that only repaired DVCs when packets arrived on them.

Factors 1 and 3 above are no longer issues, so it is reasonable to expect that smaller buffers could be used. It is important to have a strategy that addresses

6.1.3 Asynchronous System Design

Prior to implementing AIR (the routing algorithm for the CMF), I constructed a flow chart depicting the anticipated state transitions. This was the basis for design of the routing code. As I have repeatedly experienced (but apparently consistently failed to learn), designing asynchronous systems is very difficult—and it is even more difficult in an unreliable environment.

Consider the *happened-before* relationship (denoted: →) defined by Leslie Lamport:

- If events a and b occur in the same process, a→b if a preceded b.
- If a is the sending of a message and b is the reception of the message, a→b.
- If a→b and b→c then a→c.

I and many other developers of distributed systems tend to subconsciously add a fourth lemma:

- If a→b and a→c and b→d, then c→d.

In other words, if a process X sends messages m1 and m2 to Y, Y will receive m1 before m2. Unfortunately, this is not always the case. I spent a significant amount of time debugging the AIR protocol implementation to take into account situations where messages arrived out of order and/or arbitrary messages were dropped. If I were to do the implementation over again, I would invest (either money or development time) in a asynchronous state machine generator that takes a protocol as its input and generates a state machine that characterizes every possible state that the protocol can be in, with stubs into which code to handle those states is inserted.

6.2 Performance Evaluation

6.2.1 The Fidelity of MANE and Virtual Machines

An persistent problem in the evaluation of CMF has been the breaking of DVC connections due to transient topology changes. As was described above, the CMF uses packet acknowledgements both to ensure reliable packet delivery and to quickly detect topology changes. This design levies serious requirements on the fidelity of the emulation test bed. If the test bed were to intermittently fabricate transmission latencies that exceed the CMF's acknowledgement timeout several times over, the CMF would interpret this as a no-longer-in-range neighbor and would break / re-route the DVCs that traverse that link. Well, this is exactly what was happening in the evaluations. The latency between transmitting a packet and receiving the acknowledgement for that packet, which is usually measured in single-digit milliseconds, occasionally took over half a second.

The BAE wireless emulation test bed is based on software developed by NRL. MANE (Mobile Ad-hoc Network Emulator) is the successor to MNE (Mobile Network Emulator). One of the significant changes made to the system was to alter the communication architecture. Both systems utilize a tun/tap tunnel to control which hosts receive a packet. MNE places the packet control at every host. The nodes are configured with virtual devices by which packets are transmitted and received. Outgoing packets are turned into broadcasts and tunneled to the physical device. Every node, upon receipt of a packet, decides whether they are out of range of the sender (dropping the packet) or in range (sending the packet through the tunnel to the virtual device).

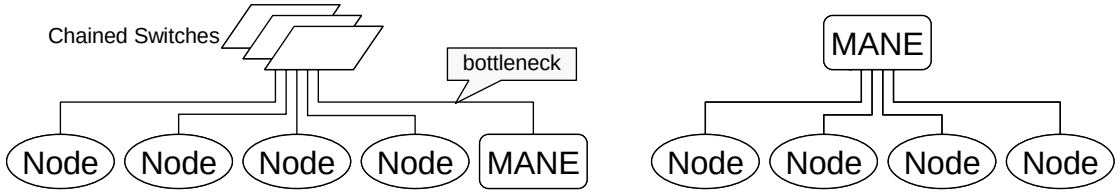
MANE takes s different approach. It utilizes a central server (*the MANE server*) that is responsible for determining which nodes in the network can hear a given packet. Every host still has an internal virtual device for sending and receiving packets. However, the virtual device now translates every outgoing packet to a unicast directed to the MANE server. The server decides which hosts are to receive the packet, and then unicasts the packet to each of those hosts.

In a dense topology, every transmitted packet may be received by twenty or more hosts. I run a 60-node MANET, where each host was originating six ~450-byte data packets a second. If we assume that the average distance from source to destination is two hops, then each host was sending twelve of these packets a second. Many of the acknowledgment packets acknowledged multiple data packets, so assume that each host also sent six ~50-byte acknowledgement packets a second. This results in a load on the Ethernet link between the switch and the MANE server of:

$$60 \times (12 \times 450 + 6 \times 50) \times 21 \times 8 = 57.5 Mb$$

This is very close to the knee of the performance curve for a 100 Mb Ethernet line, and so it is not surprising that I would occasionally see packets experiencing excessive delays.

NRL avoids this problem by using specially-configured hosts as their MANE servers. They place multiple 4-port NIC cards into powerful Linux servers and chain the servers together, allowing them to construct a switch-less hub-and-spoke topology, where every node in their test bed has a dedicated Ethernet connection to the server (using a cross-over cat5 cable) (Figure 5). While the NRL approach prevents there being a bottleneck, it is stunningly expensive (consider that every fourth time one adds a host to the network, one has to buy a new 4-port NIC, and every 16th host requires a new (and expensive) server to add to the MANE server cluster.



BAE Test Bed Network Architecture *NRL Test Bed Network Architecture*

Figure 5: The difference between the BAE and NRL network topologies.

Another problem that we have seen is associated with virtual machines (VMs). We use VmWare® on our test range to multiply the number of hosts. To minimize performance impact, I ran my simulations with only a single VM per physical host (I had to run within a VM due to the security plan in place for the network; this constraint will hopefully be removed in the future). However, we have observed VMs pausing for tenths of a second at a time for no apparent reason. If two interacting VMs were to pause for 0.3s at just the wrong points in time, that would cause the 0.6s latency packet round trip that I observed in the system.

I have not done enough of an investigation to allocate the blame for the occasional extreme transmission latencies to these two factors. When I recommence performance evaluations, I will attempt to mitigate these factors by a) crafting my own MANET movement models that are more sparse (i.e., each node has fewer one-hop neighbors) and b) use an application that exhibits more locality (see the next section). I am hopeful that I can convince the folks working on the T-AIDR project (the project that "owns" the test bed) to investigate and hopefully cure these performance issues.

6.2.2 The Necessity of a Holistic Approach

In the proposal for the CMF Program, I argued that a fault in previous MANET research efforts was the *reductionist* approach to their investigations. Specifically, researchers would take a system designed for conventional wired networks and apply them to a MANET, while changing a single component to measure the degree to which the new version of the component improved MANET performance. To be fair, reductionism (understanding a complex system by isolating and then investigating its components) is the classic scientific approach to studying complex systems. The problem posed by MANETs is that the phenomena that are of interest to MANET designers (high packet latencies, low throughput, the high incidence of dropped packets, limited system scalability) are the result of many inter-related and interacting factors. A MANET's complexity stems not simply from the fact that there are many subsystems within the MANET, but also from the fact that these subsystems interact with each other in subtle ways. A MANET is a chaotic, asynchronous multi-agent system.

This presents the researcher with two problems. The first is that the performance objective that one is trying to achieve is rarely under sole influence of the component one is affecting. Thus, one can make significant changes to the behavior of a component without seeing any change in the performance of the system as a whole. The second problem is that, in order to be able to observe the MANET as a whole (and to make the research logistically feasible), researchers typically work in a simulation. Simulations are, by their nature, simplifications of reality. When a system is put into simulation, the scientist will implement with high fidelity the first-order effects that drive the behavior being examined, leaving out or simplifying

the second order effects that are he determines to be less significant in determining the outcome. Unfortunately, MANETs are chaotic systems; these “second order effects” can in fact play a fundamental role in determining network behavior. This was shown in [1], where Dr. Gray observed not only a quantitative shift in MANET performance from simulation to reality, but also a qualitative reordering; the protocols that behaved worse in simulation outperformed their competitors when operating in a real MANET.

So, I knew going in that a holistic/systemic approach to MANET research was necessary. I also knew going in that there are some fundamental principles that a MANET must observe in order to be viable, and that amongst these is *topological locality*. So, what did I do? I implemented a toy application to exercise the MANET that does not have locality as a property. The *Random Crawl* application exercises the principal of *discovery and self-organization*; each thread selects the next destination to communicate with by asking its current communication partner for the address of a node that it is currently communicating with. This process is bootstrapped via the node’s two-hop neighborhood. In other words, when a thread first starts up, it selects a node from within its two-hop neighborhood and begins to send it a packet stream. That node responds with the identity of a node in its local topology, which comprises its two-hop neighborhood plus all nodes that are on DVCs that traverse that node. Thus, the thread “crawls” across the MANET, selecting communication partners that are potentially further and further away from it. And as the node’s DVCs grow in length, its local topology grows in size, allowing it to provide next hop addresses to nodes communicating with it that are further and further away. Voilá! No locality.

The effect of this oversight on my part can be seen in Figure 6. This graph compares the performance of AIR and OLSR as the network scales. Specifically, this graph is examining the overhead of the two protocols. To examine overhead, we measured the total number of bytes sent/received on each radio. So, the graph shows both the total number of random crawl packets successfully delivered and the total amount of data seen on the radios. You will quickly note that the “overhead” grows explosively, and nearly linearly, as the network grows in size. You will also note that it grows equally for OLSR and for AIR. This is because almost 100% of the “overhead” being received on radios is random crawl data packets. This is a factor both of the density of the MANET being examined (I used a movement model provided by the T-AIDR FCS test team, which is very dense) and the number of hops being traversed by the packets.

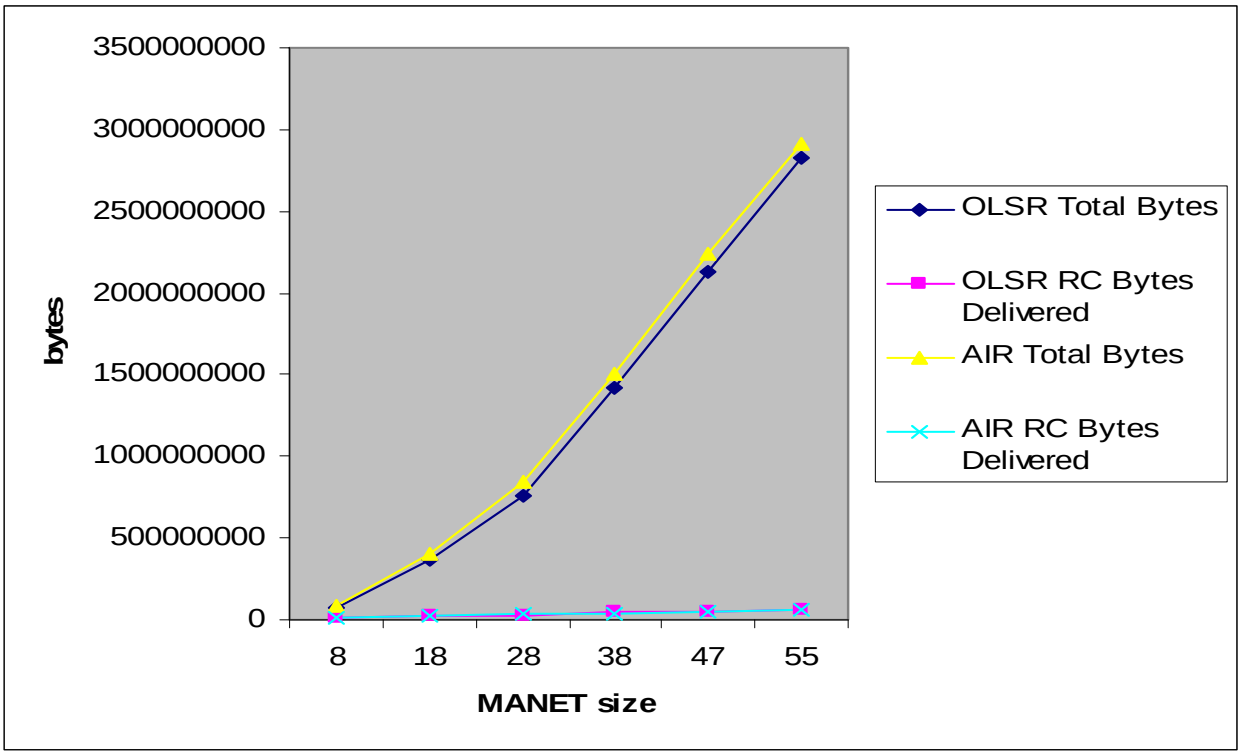


Figure 6: How overhead scales with MANET size.

Note that the overhead is dominated by over-heard data packets.

Lesson learned – I cannot ignore topology and locality when doing an evaluation, because either of these attributes can dominate the observed performance.

7 Results

As indicated in the graph above (Figure 6), the comparison of overhead between AIR and OLSR was flawed. However, we did successfully compare the ability of the two protocols to deliver packets in highly dynamic environments. Figure 7 shows the movement model used to measure the ability of the two protocols to deliver packets in dynamic environments.

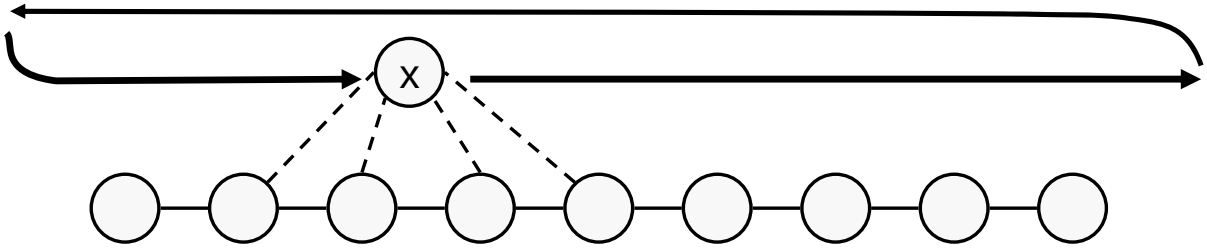


Figure 7: The dynamic network movement model. Nine nodes in a "chain of pearls", with a tenth node moving back and forth in front of them.

Table 1 shows the packet delivery figures for a ten-node MANET running the movement model in Figure 7. The results for AIR are typical of its performance, independent of the movement model. The dropped packets appears to be primarily due to intermittently dropped links (a problem described in §6.2.1), but more study is required. OLSR, on the other hand, displays significant problems with packet delivery. The

Table 1: Performance results for a dynamic network.

	AIR	OLSR
Total data packets sent	29,766	29,766
Total data packets recd	28,136	24,064
Data packets recd by node *x*	2,543	1,779

movement of node *x* appears to not only confound OLSR's ability to deliver packets to/from *x*, but it also impacts the ability to deliver packets between the other nodes (note that over 5,000 packets were dropped, but only ~2,000 packets to/from *x* were dropped). This means that a) OLSR was slow to respond when links to/from *x* were dropped and b) OLSR established routes between other packets that traversed *x*.

7.1 Comparison to Planned Objectives

The Common MANET Framework (CMF) was a research and development project that targeted three separate aspects of MANET development. First, it examined the design patterns for robust MANET applications and the APIs that the MANET must provide to support those design patterns. Second, it explored a novel routing mechanism, called *Application-Informed Routing*, that utilizes a symbiotic relationship between the *application layer* (e.g., middleware plus user applications) and the *routing layer* to implement responsive, scalable MANET routing. Third and finally, the CMF constructed a standard interface to the transmission layer, allowing the CMF to be *bound* to multiple transmission technologies, possibly simultaneously, resulting in a capability for heterogeneous MANET deployment.

At this point, we are content with the API design and construction. The true test will be when AFRL middleware is integrated with the CMF, and we anticipate that there will be requests for modifications and/or additions to the API set. However, we have performed paper-and-pencil integration of a number of applications that are desirable in the tactical environment, including VoIP and situational awareness, and have implemented toy applications such as the MANET Browser and RandomCrawl, as well as constructing a large number of command-line utilities for enumerating the contents of the known topology, examining the membership of individual dynamic virtual circuits (e.g., the path to a communication partner), etc. A key late addition to the API set (and associated CMF services) is a statistics gathering and reporting framework. The statistics gathering module is executed once per second, maintaining a 60-second history of the state of the node. APIs then query these statistics, allowing a middleware developer to construct a daemon to monitor a node's state (and possibly to share that state with neighboring nodes).

AIR – the routing algorithm – appears to be meeting expectations. The implementation is successful, and dynamic virtual circuits (DVCs) perform admirably in maintaining routes across the MANET. In comparisons with OLSR, we have demonstrated that AIR is far more responsive in highly dynamic environments. Of course, being responsive to network dynamics makes AIR sensitive to timing variances in the emulation environment used for testing – we are having real problems achieving the timing fidelity necessary for the CMF to behave properly. Right now, this appears to be a minor issue, not seriously impacting the performance of the CMF in our performance evaluations, but it is still bothersome.

The Transmission Layer Interface (TLI) has successfully bound the CMF to wired networks (both using broadcast and multicast bindings), to the BAE wireless emulation test bed, and to 802.11 radios. That being said, the TLI is the component with measurable development left undone. Specifically, a raw socket binding was not successfully integrated with the CMF before the end of the contract, and the binding configuration does not currently support multiple devices simultaneously. These features could be completed within a week, but testing them would require at least another week, and ultimately we chose to focus on stabilizing the current version of the CMF and obtaining performance measurements instead of performing this development.

7.2 Technical Accomplishments

If we define *technical accomplishments* as publishable results, then there are four central accomplishments. The first is AIR as a unicast routing protocol. The second is the AIR multicast routing protocol. It has not received the focus that the unicast protocol has to date, and yet it is perhaps a more significant achievement. The third is a mathematical model for the cost of not performing hop-level acknowledgements, with an associated argument that the CMF, by virtue of implementing hop-level acknowledgements, transmission retries and immediate re-routing in response to network dynamics, has the ability to support TCP over a MANET. The fourth accomplishment is an understanding of how to construct robust, scalable MANET applications.

Our performance evaluations and the paper we have written for MILCOM ’08 are focused on the first accomplishment. Our hope is to find the means to continue the performance evaluations and to publish papers on at least two of the other three accomplishments.

7.3 Technology Transition and Transfer Description

The source code, make files and all documentation have been delivered to AFRL. The early alpha versions of the software were quite flaky. The latest version of the software successfully executes across a highly-connected 54-node MANET transmitting excessive amounts of application traffic. Valgrind was used to eliminate memory leaks and to assist in finding pointer mismanagement. Unfortunately, the wireless test bed is operated in a classified laboratory, which means that bug fixes performed on the test bed had to be transcribed, brought out of the laboratory as hardcopy, and then hand-entered back into the system. I have already found one case where I fat-fingered code modifications – I do fear that I will receive bug reports that were fixed in the test bed but not in the code base.

If AFRL contracts approves the release, we will move the code base into the public domain via an open-source license (the GNU Lesser General Public License).

7.4 Publications

A paper has been written for submission to MILCOM ’08, conditional on AFRL confirming that the CMF program is a 6-1 research project (and, thus, exempt from ITAR restrictions). Unfortunately, we will not receive permission in time to submit to MILCOM. However, we will complete the paper as a tech report, and submit it to the next appropriate conference when we receive permission to do so. As alluded to above, we hope to publish additional papers based on the work on this program.

7.5 The Software

The CMF is delivered as a single directory tree whose root is CMF_x.y.z/, where x.y.z is the version number. Beneath that directory, the software is partitioned into modules, with each module receiving its own subdirectory. Each module possesses its own include directory and is built separately. The modules:

packets	Contains routines for building, parsing and printing packets. The specification for the CMF packets is in the include directory for this module, in the file cmf_packet_headers.h.
routing	The routines for handling the six types of DVC packets (creation, teardown, route established, invalid route, IP and discovery).
local_ip	This is the interface to the local host. It includes a thread that reads packets from the CMF tun device and a function for writing packets to the tun device.
services	This package contains all of the CMF services, and is thus rather overloaded. The services support/manage: the known topology, dynamic virtual circuits, discoveries, multicast destinations, and client connections (a client is an application or middleware connecting via libcmf.so or cmf.jar).

tli	The elements involved in transmitting packets to neighboring nodes. Key players are datapkt.c (which is the packet buffer – it sits between the router and the TLI), device.c (which tracks neighbors and which device they can be reached on), transmission_thread.c (which chooses between announcements, acknowledgements and DVC packets to transmit) and recv.c.
util	Functions that are common across the CMF. The key player here is node.c, which is the CMF representation of an IP address. Note that it can be either an IPv4 address or an IPv6 address – or any other sequence of bytes.
bindings	These bind the TLI to specific transmission technologies.
unclass_testbed	Configuration files for running experiments.
api	The code for building libcmf.so and cmf.jar.
apps	Contains the applications that were developed in conjunction with the CMF. RandomCrawl is notable as forming the basis for the performance evaluations to date.
include	Contains *.h files common across the CMF project. cmf.h is installed in /usr/include – it is the declarations for libcmf.so. cmf_build_parameters.h specifies the configuration parameters that control how the CMF performs. And node.h is needed, because the functions in libcmf.so whose arguments or outputs are IP addresses take them in the form of Node structures.
bootup	Routines placed in /etc/init.d or in /usr/sbin that support launching the CMF.
man	Man pages for the CMF (not well maintained – we recommend looking at the comments in cmf.h for API reference).

7.6 Installing and Running the CMF

The CMF builds on any (as far as we know) Linux kernel that supports tun/tap devices. Our test platforms have been Fedora Core 6, 7 and 8. When using a radio, we have consistently run the CMF in a virtual machine running Linux, with the host running Windows XP and the virtual machine's network interface bridged.

We have tested the CMF using two network configurations. In the first, the host has only a single network interface named "eth0". In the second, the host has multiple network interfaces, with at least one of them named "manet." The following installation procedure works with either of these cases.

To install the CMF, cd to the top-level CMF directory, then type "make; make install; cd apps; make; make install". This will build and install the CMF and the applications provided with the CMF.

Installing the CMF does the following:

- It places the cmf executable into /usr/sbin, along with the utility programs initialize_tun, ip_chooser.sh and broadcast_ip_chooser.sh.
- cmf.h is placed into /usr/include.
- cmf.init is copied to /etc/init.d/cmfinit. This has some unfortunate ramifications that will be discussed.
- Many command-line utilities are installed into /usr/bin.
- libcmf.so and cmf.jar are installed into /usr/lib.

To remove the CMF, cd to the top-level CMF directory and type "make uninstall; cd apps; make uninstall".

To launch the CMF, type “service cmfinit start”. It would be great if the service were “cmf,” or even “cmfd.” But it isn’t. You can verify that the CMF is running either by invoking **`ps –ef | grep cmf`** or **`/usr/bin/getTopology`**.

7.7 How to add an API

There are many things that a developer may wish to do with the CMF, but we anticipate that the most common thing will be to add a new API. A short description of how to do this will both dramatically shorten the time needed to figure it out and will help one to understand the CMF’s architecture. Perform the following steps:

1. Modify cmf.h to include your new API. Try to follow the naming conventions already in place.
2. If you are adding a new command, edit topology_protocol.h in cmf_x.y.z/include. This is a badly named file – it actually sets the numeric value that represents each command when a client sends the command to the CMF services.
3. Edit cmf_x.y.z/api/src/topology_client.c to implement the function that was defined in cmf.h. Again, a badly named file; the routines here are not necessarily topology specific. Note that there is also a discovery_client.c – please do not add routines there. Some day that file will go away. Note that editing topology_client.c is tricky. The functions in this file serialize their arguments into a buffer and then use a local socket to send the request to the socket handler (see the services package). They then receive a response from the local socket and present the response to the caller in an appropriate format.
4. Edit the socket handler, if necessary (it probably is not). The socket handler looks at the 4 most-significant bits of the command identifier, and then routes the request to the correct protocol handler.
5. Edit one of the protocol handlers (e.g., topology_protocol_handler.c, dvc_protocol_handler.c, etc.).
6. Voila, you now have a new CMF API function.

7.8 Miscellaneous Notes

The current binding wraps CMF packets inside of UDP broadcasts. So, counting the packet layers, you have: data | UDP | IP | CMF DVC | CMF 1-hop | UDP | IP. Given they way that routing is set up under Linux, this means that the CMF tun device has one IP address while the Ethernet port has a different IP address. The automated install process replaces the first byte of the host’s IP address with ‘11’. Thus, if eth0 is bound to 10.10.92.16 with netmask 255.0.0.0, the CMF will see the host’s IP address as 11.10.92.16, and the binding will wrap CMF packets inside of UDP packets addressed to 10.255.255.255. It would be better if the binding used a raw socket, and simply wrapped the CMF packet inside of an Ethernet header with a destination address of FF:FF:FF:FF:FF:FF.

The first time the CMF is launched on a host, it creates a configuration file /etc/sysconfig/cmf, and further a configuration directory /etc/sysconfig/cmf.d. The contents are self-explanatory.

8 References

[1] R. S. Gray, D. Kotz, C. C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, and S. McGrath, “Outdoor Experimental Comparison of Four Ad Hoc Routing Algorithms,” In *Proceedings of the Seventh ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2004),* Venice, Italy, October 2004.

[2] D. Kotz, C. C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott, "Experimental Evaluation of Wireless Simulation Assumptions," In *Proceedings of the Seventh ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2004)*, Venice, Italy, October 2004.

[3] J. Liu, Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone, "Empirical Validation of Wireless Models in Simulations of Ad Hoc Routing Protocols," *Simulation: Transactions of The Society for Modeling and Simulation International*, 81(4):307-323, April 2005.

[4] S. Nanda and R. S. Gray, "Multipath Location Aided Routing in 2D and 3D," In *Proceedings of the Sixth IEEE Wireless Communications and Networking Conference (WCNC 2006)*, Las Vegas, NV, April 2006.

[5] R. S. Gray, "Soldiers, Agents and Wireless Networks: A Report on a Military Application," In *Proceedings of the Fifth International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents (PAAM 2000)*, Manchester, England, April 2000.

[6] P. Jacquet, P. Mühlethale, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol," In *Proceedings of the Fifth IEEE National Multi-Topic Conference (INMIC 2001): Technology for the 21st Century*, Lahore, Pakistan, December 2001.

[7] J. P. Macker, J. Dean, and W. Chao, "Simplified Multicast Forwarding in Mobile Ad Hoc Networks," In *Proceedings of the 2004 Military Communications Conference (MILCOM)*, Monterey, CA, October 2004.

[8] S.-J. Lee, W. Su, and M. Gerla, "On-Demand Multicast Routing Protocol in Multihop Wireless Mobile Networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, 7(6):441-453, December 2002.

[9] N. Arad and Y. Shavitt, "Minimizing Recovery State in Geographic Ad-Hoc Routing," In *Proceedings of the Seventh Annual ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, Florence, Italy, May 2006.

[10] G. Xing, C. Lu, R. Pless, and Q. Huang, "On Greedy Geographic Routing Protocols in Sensing-Covered Networks," In *Proceedings of the Fifth Annual ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2004)*, Tokyo, Japan, May 2004.

[11] B. Karp and H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks," In *Proceedings of the Sixth Annual ACM/IEEE Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, August 2000.

[12] A. Iwata, C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, "Scalable Routing Strategies for Ad hoc Wireless Networks," *IEEE Journal on Selected Areas in Communications*, 17(8):1369-1379, August 1999.

[13] Y. Ko and N. Vaidya, "Location-Aided Routing in Mobile Ad Hoc Networks," In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)*, Dallas, TX, August 1998.

[14] C. E. Perkins and E. M. Royer, "Ad Hoc On-Demand Distance Vector Routing," In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 1999)*, New Orleans, LA, February 1999.

[15] J. Li, J. Jannotti, D. DeCouto, D. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad-hoc Routing," In *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, August 2000.

[16] Y. Wang and X.-Y. Li, "Distributed Low-Cost Backbone Formation for Wireless Ad Hoc Networks," In *Proceedings of the Sixth Annual ACM International Symposium on Mobile Ad Hoc N etworking and Computing (MobiHoc 2005)*, Urbana-Champaign, IL, May 2005.

[17] E. M. Belding-Royer, "Multi-Level Hierarchies for Scalable Ad Hoc Routing," *Wireless Networks*, 9(5):461-478, September 2003.

[18] R. Flury and R. Wattenhofer, "MLS: An Efficient Location Service for Mobile Ad Hoc Networks," In *Proceedings of the Seventh Annual ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, Florence, Italy, May 2006.

[19] I. Abraham, D. Dolev, and D. Malkhi, "LLS: A Locality Aware Location Service for Mobile Ad Hoc Networks," In *Proceedings of the DIAM M-POMC Joint Workshop on Foundations of Mobile Computing*, Philadelphia, PA, October 2004.

[20] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers," In *Proceedings of the ACM SIGCOMM 1994 Conference on Communications Architectures, Protocols and Applications*, London, UK, August 1994.

[21] D. B. Johnson and D. B. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," In T. Imielinksi and H. Korth, editors, *Mobile Computing*, Kluwer Academic Publishers, 1996.

[22] V. Ramasubramanian, Z. Haas, and E. G. Sirer, "SHARP: A Hybrid Adaptive Routing Protocol for Mobile Ad Hoc Networks," In *Proceedings of the Fourth Annual ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2003)*, Annapolis, MD, June 2003.

[23] A. Srnivas, G. Zussman, and E. Modiano, "Mobile Backbone Networks – Construction and Maintenance," In *Proceedings of the Seventh Annual ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2006)*, Florence, Italy, May 2006.

[24] Y. Tamir and Y. F. Turner, "High-Performance Adaptive Routing in Multicomputers Using Dynamic Virtual Circuits," *6th Distributed Memory Computing Conference*, Portland, OR, pp. 404-411 (April 1991).

[25] Wendi Heinzelman, Joanna Kulik, and Hari Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks," *Proc. 5th ACM/IEEE Mobicom Conference*, Seattle, WA, August 1999.

[26] G. Turi, M. Conti, and E. Gregori, "A Cross Layer Optimization of Gnutella for Mobile Ad Hoc Networks," In *Proceedings of the Sixth Annual ACM International Syposium on Mobile Ad Joc Networking and Computing (MobiHoc 2005)*, Urbana-Champaign, IL, May 2005.

[27] Y. Yang, X. Wang, S. Zhu, and G. Cao, "SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks," In *Proceedings of the Seventh Annual ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2007)*, Florenace, Italy, May 2006.

[28] H. Zhou and S. Singh, "Content Based Multicast (CBM) in Ad Hoc Networks," In *Proceedings of the First ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2000)*, Boston, MA, August 2000.

[29] A. K. Pandey and H. Fujinoki, "Study of MANET Routing Protocols by GloMoSim Simulator," *International Journal of Network Management*, 15(6):393-410, November/December 2005.

[30] *The GNU Radio Project*, http://www.gnu.org/software/gnuradio/.

[31] *Java™ Platform, Standard Edition 6 API Specification*, http://java.sun.com/javase/6/docs/api/.

[32] "FAST Waveform to be Integrated into MIDS," *Defense Update*, http://www.defense-update.com/newscast/0107/news/310107_fast.htm.